Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 55 (2008) 1660–1669
An International Journal
computers & mathematics
with applications
www.elsevier.com/locate/camwa

Languages of logic and their applications

K. Pásztor Varga[a,*], M. Várterész[b]

[a] *Department of Programming Languages and Compilers, Eötvös Loránd University, H-1117 Budapest, Pázmány Péter sétány 1/C., Hungary*
[b] *Faculty of Informatics, University of Debrecen, H-4010 Debrecen, P.O.Box 12., Hungary*

Received 15 May 2007; accepted 7 June 2007

Abstract

Concerning the logical description languages, in the past 40–50 years many authors have introduced a number of structurally very different first-order languages. Some of these languages follow the structure of a given future model, other ones have been prepared for the description of an arbitrary model. Other variations of the first-order languages do not follow the whole structure of any model: they have been prepared only for the relations definable over the universe in order to be able to prove the generalizations of a number of difficult logical results.

The semantics of the first-order languages is based on the interpretation of their extralogical symbols by a suitable model. In some cases, in the interpretation all possible models can be in focus, but there are cases when the models over a special universe are regarded. The naming problem of the universe element of the model arises at this stage. The efforts for solving this problem lead to different approaches.

Here, we present the most important language definitions and some characteristic semantics. We investigate the different approaches and conclude that they do not indicate essential differences. In fact, they have been only motivated by seeking for an easier way to achieve the just fixed target. Moreover, we try to point out the suitability connections of languages and semantics definitions.

Keywords: First-order languages; Syntax; Semantics

1. Syntax

Leibniz (1640–1710) was the first, who brought up the idea of a complete formal logical reasoning system. He tried to develop a language, and a calculus of reasoning, called them the "lingua characteristica" (universal language) and the "calculus ratiocinator" (calculus of reasoning). Leibniz's work in this area was basically unknown till its publication in [1], so his ideas were prescient but not influential. Frege developed the base of the modern logical grammar in his book [2] in 1879. He gave the first formal treatment of logic including both quantifiers, relation symbols and propositional connectives. Frege gave, also for the first time, the definition of a proof as a finite sequence of formulae, each of which is either an axiom or follows from previous formulae of the proof by an application of a rule of inference. The appreciation of Frege's works is also posterior probably because of their hard readability.

* Corresponding author.
E-mail addresses: pkata@ludens.elte.hu (K. Pásztor Varga), varteres@inf.unideb.hu (M. Várterész).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2007.06.007

In 1889 Peano published a paper [3] similar to Frege's conception, but his notation system was quite other. In contrast with Frege's papers, this notation system became known fast and widely used. Roughly speaking, Peano's notation extended and modified by Russell and Hilbert is used today. Moreover, forgetting that Frege was the first who applied this calculus, it is called Hilbert style in the literature. In the 1920s, Löwenheim and Skolem observed that function symbols and constant symbols (prospective names for elements of a domain) may be useful for formal treatment of logic. In effect, constructions from these items make sets of terms into the first-order logic languages.

In fact, development and publications of different versions of the first-order logic languages in current use have some periods. In the first general works after 1930s [4–6] the system of extralogical symbols of first-order languages took shape from the signs of (mathematical and logical) functions. The logical components of logic languages are common (connectives, quantifiers and a countable set of (individuum) variables). Now, we give a formal definition of what such a language constitutes.

Definition 1.1. The alphabet of a first-order language consists of

(i) logical symbols:
  (1) connectives and quantifiers: $\neg, \wedge, \vee, \supset, \forall, \exists$,
  (2) variables: $x_1, x_2, x_3, \ldots$,
(ii) extralogical symbols:
  (1) for each natural number n, named n-ary predicate symbols: $P_1^n, P_2^n, P_3^n, \ldots$,
  (2) for each natural number n, named n-ary function symbols: $f_1^n, f_2^n, f_3^n, \ldots$,
  (3) constant symbols: $c_1, c_2, c_3, \ldots$ (the constant symbols may be simply listed as 0-ary function symbols $f_1^0, f_2^0, f_3^0, \ldots$),
(iii) punctuation: ')', '(' and ','.

The object of study in mathematics is frequently a set together with a structure defined on it. For example, the set of triangles with similarity relations, the set of real numbers with the operations of addition and multiplications, and so on. A more precise definition of this concept has been introduced by the next definition of a formal system. The formal system is a tuple $\langle \mathcal{U}, \mathcal{R}, \mathcal{M}, \mathcal{C} \rangle$ where $\mathcal{U}$ is a nonempty set, $\mathcal{R}$ is a finite set of relations on $\mathcal{U}$, $\mathcal{M}$ is a finite set of operations on $\mathcal{U}$, $\mathcal{C}$ is a finite (possibly empty) set of distinguished elements on $\mathcal{U}$.

Then, the formal systems have been characterized with signatures. A signature μ is a mapping that associates some natural number called arity to every relation and operation. The arity gives the number of arguments of relations or operations. Thus, the formal system is a quintuple $\langle \mathcal{U}, \mathcal{R}, \mathcal{M}, \mathcal{C}, \mu \rangle$. The description languages of the formal systems appear with signatures in the form $\langle \mathcal{R}^*, \mathcal{M}^*, \mathcal{C}^*, \mu \rangle$ where the elements of $\mathcal{R}^*, \mathcal{M}^*, \mathcal{C}^*$ are names of the elements of $\mathcal{R}, \mathcal{M}, \mathcal{C}$ and μ is the associated signature.

Example 1.1. The arithmetic as a formal system is the quintuple $\langle \mathbf{N}_0, \mathcal{R}, \mathcal{M}, \mathcal{C}, \mu \rangle$. The description language is the tuple $\langle \{\leq\}, \{\mathbf{s}, +, \times\}, \{\mathbf{0}\}, \mu \rangle$ where

– $\leq$ is the name of the only relation in $\mathcal{R}$,
– $\mathbf{s}, +, \times$ are the signs of the operations in $\mathcal{M}$,
– $\mathbf{0}$ identifies the smallest element of the universe $\mathbf{N}_0$,
– and the signature is the following:

$\mathcal{R}$	$\mu(\mathcal{R})$	$\mathcal{M}$	$\mu(\mathcal{M})$
$\leq$	2	$\mathbf{s}$	1
		$+$	2
		$\times$	2

The effect of the above mentioned approach appears later in the various definitions of the first-order languages. As in Definition 1.1, these languages contain three parts, (i) logical symbols, (ii) extralogical symbols and (iii) punctuation. The extralogical symbols vary from language to language, while the items (i) and (iii) are common to all languages.

Definition 1.2. A first-order language of this kind is determined by specifying

(ii) the sets of extralogical symbols
  (1) $\mathcal{P}$: a nonempty set of predicate symbols,
  (2) $\mathcal{F}$: a set of function symbols,
  (3) $\mathcal{C}$: a set of constant symbols,

and a signature μ that consists of mappings $\mu_{\mathcal{P}}$ and $\mu_{\mathcal{F}}$, where

– $\mu_{\mathcal{P}} : \mathcal{P} \to \mathbf{N}$ gives the arity of every predicate symbol,
– $\mu_{\mathcal{F}} : \mathcal{F} \to \mathbf{N}$ gives the arity of every function symbol.

We use the notation $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$ for the first-order language determined by the sets $\mathcal{P}, \mathcal{F}, \mathcal{C}$ with signature μ. Even now, it is usual to think of constant symbols as 0-ary function symbols. In these cases, a first-order language is a triple $\langle \mathcal{P}, \mathcal{F}, \mu \rangle$. The symbol sets $\mathcal{P}$, $\mathcal{F}$ and $\mathcal{C}$ may be finite or infinite and, except $\mathcal{P}$, they may be even empty. Let us mention that

- in works [7–10] the sets of predicate and function symbols are finite. These languages contain
  (1) a finite nonempty set $\mathcal{P}$ of the predicate symbols $P_1, P_2, \ldots, P_k$ $(k \geq 1)$,
  (2) a finite set $\mathcal{F}$ of the function symbols $f_1, f_2, \ldots, f_l$ $(l \geq 0)$,
  (3) a finite or countable set $\mathcal{C}$ of the constant symbols $c_1, c_2, \ldots$ and a signature μ sometimes designated as

$$\begin{pmatrix} P_1 & P_2 & \cdots & P_k & ; & f_1 & f_2 & \cdots & f_l \\ n_1 & n_2 & \cdots & n_k & ; & m_1 & m_2 & \cdots & m_l \end{pmatrix}.$$

- Certain authors [11–15] work with infinite sets of predicate and function symbols. In [16] Ershov and Palutyin allow finite and infinite sets, as well.

Finally, it should be mentioned that Smullyan [17] also developed a definition for the first-order languages. His language does not contain any function and constant symbol, it includes only the so-called parameter symbols instead.

Definition 1.3. The Smullyan's version of the first-order languages consists of logical symbols (i), extralogical symbols (ii) and punctuation (iii).

(ii) The extralogical symbols are determined by a pair $\langle \mathcal{P}, Par \rangle$, where
  (1) $\mathcal{P}$ is a countable list of n-ary predicate symbols for every natural number,
  (2) *Par* is a countable list of symbols called parameters.

The role of parameter symbols is quite different from constant symbols.

Having specified the basic element of syntax, the alphabet, we go on to grammar rules of the languages. The definition can be formulated for all logic languages in a common way. We specify the expressions (terms and formulae) of a first-order language by inductive definitions which select certain "well-formed" strings of symbols, exactly those we take as meaningful ones. It is obvious that only the symbols of the given logic language appear in the next grammar rules effectively.

Definition 1.4 (*Terms*).

(i) Any variable, any constant symbol and any parameter symbol is a term.
(ii) If f is an n-ary function symbol and $t_1, t_2, \ldots, t_n$ are terms, then $f(t_1, t_2, \ldots, t_n)$ is a term too.
(iii) A string is a term only in the case if it can be constructed by finitely many applications of the rules (i)–(ii).

Definition 1.5 (*Formulae*).

(i) If P is an n-ary predicate symbol and $t_1, t_2, \ldots, t_n$ are terms, then $P(t_1, t_2, \ldots, t_n)$ is an (atomic) formula.
(ii) (1) If A is a formula so is $\neg A$.
  (2) If A and B are formulae, so are $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$.
  (3) If A is a formula and x is a variable, then $\forall x A$ and $\exists x A$ are formulae.
(iii) A string is a formula only if it can be generated by finitely many applications of the rules (i)–(ii).

We distinguish free and bound occurrences of variables. An occurrence of a variable x in a formula A is bound if there is a subformula of A containing that occurrence of x such that it begins with $\forall x$ or $\exists x$. An occurrence of x in A is free if it is not bound. Additionally, an expression is called closed if no variable has free occurrence in it.

2. Semantics

In [2] Frege gave the concept of quantifiers as ranging over all objects. The model in which a set is given and variables range over that given set was not introduced. In the 1890s, Schröder developed the idea of the model for the first-order logic languages. A model consists of a nonempty set, the domain or universe, together with relations and functions on this set according to relation and function symbols in the language. A first-order language with a model becomes a description language of an assigning formal system. At this stage, it becomes reasonable to ask whether some formulae are true or not in a given model.

Definition 2.1. A model for the first-order language $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$ is a pair $\langle \mathcal{U}, \mathcal{I} \rangle$ where

(i) $\mathcal{U}$ is a nonempty set, called the universe,
(ii) $\mathcal{I}$ is a mapping, called interpretation that associates
  (1) some n-ary relation $\mathcal{I}(P) : \mathcal{U}^n \to \{true, false\}$ to every n-ary predicate symbol P of $\mathcal{P}$,
  (2) some n-ary function $\mathcal{I}(f) : \mathcal{U}^n \to \mathcal{U}$ to every n-ary function symbol f of $\mathcal{F}$,
  (3) and some member $\mathcal{I}(c) \in \mathcal{U}$ to every constant symbol c in $\mathcal{C}$.

To determine the meaning of terms and formulae, we have to define the evaluation of the variables of the language. In an evaluation, the variables mean elements of the universe. Two ways of reference to the universe elements will be presented: either with a mapping $\kappa : V \to \mathcal{U}$ called an assignment or with extending the language.

Suppose, we have a model, which gives the meaning of the constant and function symbols of the language, and we have an assignment evaluating the variables. Then, we have enough information to calculate values for arbitrary terms.

Definition 2.2. Let $\langle \mathcal{U}, \mathcal{I} \rangle$ be a model for the language $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$, and let κ be an assignment in this model. To each term t of $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$, we assign a value $|t|^{\mathcal{I},\kappa}$ in $\mathcal{U}$ as follows:

(i) (1) for a constant symbol $c \in C$, $|c|^{\mathcal{I},\kappa}$ is the element $\mathcal{I}(c)$ of $\mathcal{U}$,
  (2) for a variable x, $|x|^{\mathcal{I},\kappa}$ is the element $\kappa(x)$ of $\mathcal{U}$,
(ii) $|f(t_1, t_2, \ldots, t_n)|^{\mathcal{I},\kappa} = \mathcal{I}(f)(|t_1|^{\mathcal{I},\kappa}, |t_2|^{\mathcal{I},\kappa}, \ldots, |t_n|^{\mathcal{I},\kappa})$.

This definition associates an element in $\mathcal{U}$ with each term of the language. If the term is closed its value does not depend on the assignment κ.

Now, we associate a truth value with each formula. For this, we need a preliminary notion. Let x be a variable. The assignment κ^* in the model $\langle \mathcal{U}, \mathcal{I} \rangle$ is an x-variant of the assignment κ, if $\kappa^*(y) = \kappa(y)$ for any variable y except x.

Definition 2.3. Let $\langle \mathcal{U}, \mathcal{I} \rangle$ be a model for the language $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$, and let κ be an assignment in this model. To each formula A of $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$, we assign a truth value $|A|^{\mathcal{I},\kappa}$ as follows:

(i) $|P(t_1, t_2, \ldots, t_n)|^{\mathcal{I},\kappa} = \mathcal{I}(P)(|t_1|^{\mathcal{I},\kappa}, |t_2|^{\mathcal{I},\kappa}, \ldots, |t_n|^{\mathcal{I},\kappa})$.
(ii) (1) $|\neg A|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa} = false$,
  (2) $|A \wedge B|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa} = true$ and $|B|^{\mathcal{I},\kappa} = true$,
  (3) $|A \vee B|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa} = true$ or $|B|^{\mathcal{I},\kappa} = true$,
  (4) $|A \supset B|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa} = false$ or $|B|^{\mathcal{I},\kappa} = true$,
(iii) (1) $|\forall x A|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa^*} = true$ for every assignment κ^* which is an x-variant of κ,
  (2) $|\exists x A|^{\mathcal{I},\kappa} = true$, if and only if $|A|^{\mathcal{I},\kappa} = true$ for some assignment κ^* which is an x-variant of κ.

Just as with terms, if the formula is closed then its truth value does not depend on the assignment. Moreover, the value of an expression with n free variables depends on the assignment of these variables, so its meaning in the model is an n-ary function or relation over the universe.

By the grammar, a particular language expression can contain only finite number of symbols. Thus, to specify the meaning of an expression, we have to know the interpretation only for the symbols occurring in the expression (instead of the whole language). We can say that the semantics does not mean the interpretation of the language itself, but the interpretation of symbols of the given expression. This fact makes reasonable the introduction of languages with finite predicate, function and constant symbol sets.

The following definitions based on the notion of interpretation have great importance for logic.

Definition 2.4. (i) A formula A is said to be valid, $\models A$, if $|A|^{\mathcal{I},\kappa} = \mathit{true}$ for any model $\langle \mathcal{U}, \mathcal{I} \rangle$ and any assignment κ in this model.
(ii) A set $\mathcal{S}$ of formulae is satisfiable if there is a model $\langle \mathcal{U}, \mathcal{I} \rangle$ of $\mathcal{L}$ and an assignment κ in this model so that $|A|^{\mathcal{I},\kappa} = \mathit{true}$ for every formula A of $\mathcal{S}$.

A further fundamental idea of logic is the notion of semantic consequence.

Definition 2.5. We say that a formula A is a semantic consequence of a set $\mathcal{S}$ of formulae (written $\mathcal{S} \models A$) if $\mathcal{S} \cup \{\neg A\}$ is unsatisfiable.

The semantic decision problem is to decide whether this relationship holds between $\mathcal{S}$ and A. There exists an equivalent formulation of this problem if $\mathcal{S} \neq \emptyset$.

Theorem 2.1 (*Deduction Theorem*). *Let A be a formula. Suppose B is a member of the set $\mathcal{S}$ of formulae. Then, $\mathcal{S} \models A$ if and only if $\mathcal{S} \setminus \{B\} \models B \supset A$.*

As it was shown in [7,12,14], the languages can be extended with new symbols denoting different elements of the universe to be able to describe a pre-interpretation. The introduction of symbols for naming the elements of the universe is common in the description language of some mathematical structures. For example, we can extend the description language of the arithmetics by naming the natural numbers. The name of a number can be a sequence of digits $0, 1, \ldots, 9$. We do this regardless of the successor function guarantees the referencing of natural numbers. The usage of the extended languages is comfortable in applications.

Following Girard's idea, if $\mathcal{U}$ is the universe of a model of a first-order language, we introduce a constant symbol c_u for naming each element u of $\mathcal{U}$.

Definition 2.6. A model $\mathcal{M}$ for the language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \emptyset, \mu \rangle$ consists of

(i) a nonempty set $\mathcal{U}$, the domain of the model $\mathcal{M}$,
(ii) a mapping $\mathcal{I}$, the interpretation of the model $\mathcal{M}$, that associates
  (1) to every n-ary predicate symbol P of $\mathcal{P}$, a relation $\mathcal{I}(P) : \mathcal{U}^n \to \{\mathit{true}, \mathit{false}\}$,
  (2) to every n-ary function symbol f of $\mathcal{F}$, a function $\mathcal{I}(f) : \mathcal{U}^n \to \mathcal{U}$.

We extend the language $\mathcal{L}$ to $\mathcal{L}[\mathcal{M}]$ by introducing new constant symbols c_u for all $u \in \mathcal{U}$. Then, we extend the interpretation for the new symbols: $\mathcal{I}(c_u) = u \in \mathcal{U}$ for all c_u.

Definition 2.7. First, we associate a value $|t|^{\mathcal{I}}$ with each closed term t of the extended language $\mathcal{L}[\mathcal{M}]$ as follows:

(i) $|c_u|^{\mathcal{I}} = u$,
(ii) $|f(t_1, t_2, \ldots, t_n)|^{\mathcal{I}} = \mathcal{I}(f)(|t_1|^{\mathcal{I}}, |t_2|^{\mathcal{I}}, \ldots, |t_n|^{\mathcal{I}})$.

Definition 2.8. Now, we associate a truth value $|A|^{\mathcal{I}}$ with each closed formula A of $\mathcal{L}[\mathcal{M}]$ as follows:

(i) $|P(t_1, t_2, \ldots, t_n)|^{\mathcal{I}} = \mathcal{I}(P)(|t_1|^{\mathcal{I}}, |t_2|^{\mathcal{I}}, \ldots, |t_n|^{\mathcal{I}})$,
(ii) equally as (ii) in Definition 2.3,
(iii) (1) $|\forall x A|^{\mathcal{I}} = \mathit{true}$ if and only if $|A^x_{c_u}|^{\mathcal{I}} = \mathit{true}$ for all u in $\mathcal{U}$,
  (2) $|\exists x A|^{\mathcal{I}} = \mathit{true}$ if and only if there exists an u such in $\mathcal{U}$ that $|A^x_{c_u}|^{\mathcal{I}} = \mathit{true}$.

Finally, we show the semantics of Smullyan's language $\langle \mathcal{P}, \mathit{Par} \rangle$. First, we give a nonempty set $\mathcal{U}$ called universe, then we introduce the notion of formulae with elements in $\mathcal{U}$ or more briefly $\mathcal{U}$-formulae.

Definition 2.9 (*$\mathcal{U}$-formulae*).

(i) If P is an n-ary predicate symbol and $t_1, t_2, \ldots, t_n$ are either variables or elements of $\mathcal{U}$, then $P(t_1, t_2, \ldots, t_n)$ is an atomic $\mathcal{U}$-formula.
(ii) (1) If A is an $\mathcal{U}$-formula so is $\neg A$.
  (2) If A and B are $\mathcal{U}$-formulae, so are $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$.
  (3) If A is an $\mathcal{U}$-formula and x is a variable, then $\forall x A$ and $\exists x A$ are $\mathcal{U}$-formulae.
(iii) An expression is a $\mathcal{U}$-formula only if it can be generated by the conditions (i)–(ii).

Note that, a $\mathcal{U}$-formula does not contain any parameter, moreover it is not in the original language $\langle \mathcal{P}, Par \rangle$ if only one element of $\mathcal{U}$ occurs in it.

Over a fixed universe $\mathcal{U}$, the meaning of predicate symbols of language is given by an interpretation $\mathcal{I}$ which assigns a relation $\mathcal{I} : \mathcal{U}^n \to \{true, false\}$ to each n-ary predicate symbol P of $\mathcal{P}$.

Definition 2.10. In a model $\langle \mathcal{U}, \mathcal{I} \rangle$ of the language $\langle \mathcal{P}, Par \rangle$ we can get truth values to the closed $\mathcal{U}$-formulae:

(i) $|P(u_1, u_2, \ldots, u_n)|^{\mathcal{I}} = \mathcal{I}(P)(u_1, u_2, \ldots, u_n)$.
(ii) equally as (ii) in Definition 2.3,
(iii) (1) $|\forall x A|^{\mathcal{I}} = true$ if and only if $|A_u^x|^{\mathcal{I}} = true$ for all $u \in \mathcal{U}$,
(2) $|\exists x A|^{\mathcal{I}} = true$ if and only if there is a $u \in \mathcal{U}$ that $|A_u^x|^{\mathcal{I}} = true$.

We have considered so far closed $\mathcal{U}$-formulae. Now, let $A(a_1, a_2, \ldots, a_n)$ be a closed formula containing exactly the parameters $a_1, a_2, \ldots, a_n$. Observe that, we will not fix any interpretation for the parameters. For any universe $\mathcal{U}$ and any elements $u_1, u_2, \ldots, u_n$ of $\mathcal{U}$, we obtain $A(u_1, u_2, \ldots, u_n)$ by substituting u_1 for a_1, …, u_n for a_n in the sentence $A(a_1, a_2, \ldots, a_n)$.

Definition 2.11. $A(a_1, a_2, \ldots, a_n)$ is called satisfiable if there exists at least one model $\langle \mathcal{U}, \mathcal{I} \rangle$ and at least one n-tuple $(u_1, u_2, \ldots, u_n)$ of $\mathcal{U}$ such that $|A(u_1, u_2, \ldots, u_n)|^{\mathcal{I}} = true$.

One can see that the two groups of the first-order languages are the classical languages $\langle \mathcal{P}, \mathcal{F}, \mathcal{C}, \mu \rangle$ and the language $\langle \mathcal{P}, Par \rangle$.

- The symbol system and uniform syntax make the different languages suitable for formalization of an arbitrary first-order problem, but language $\langle \mathcal{P}, Par \rangle$. Missing function symbols do not cause any problem, because an n-ary operation can be defined by an $(n + 1)$-ary relation.
- The semantics is uniform, so both the semantic properties of a formula or set of formulae and the notion of semantic consequence can be defined for every language in the same way. Similarly, the proof of deduction theorem and the drafting of the semantic decision problem do not depend on the language.

3. Naming the universe elements

Now, we show the necessity of naming the universe elements to obtain some important results in logic.

In the course of the solution of a semantic decision problem, the issue of perspicuity of all the interpretations over a given universe is raised. We can give any interpretation with the evaluation of the so-called ground atoms (closed $\mathcal{U}$-atoms) in the language $\langle \mathcal{P}, Par \rangle$. Remember, the $\mathcal{U}$-formulae, so the ground atoms are really in an extended language. Here, the interpretations can be considered as points of a field determined by all the ground atoms: a sequence of all ground atoms is called a base and an interpretation is a subsequence of the base, components of which we consider true. Interpretations determined in such form can be given by a semantic tree building on the base.

Example 3.1. Let $\langle \{P\}, Par \rangle$ be a language where P is a binary predicate symbol and let $\mathcal{U} = \{a, b\}$ be a universe. Then, $P(a, a)$, $P(b, b)$, $P(a, b)$, $P(b, a)$ is a base, $P(a, a)$, $P(b, b)$ and $P(a, a)$, $P(b, b)$, $P(b, a)$ are interpretations. The complete semantic tree based on this base is given in Fig. 1.

In case of classical languages $\langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$, when we cannot work with ground atoms since an assignment maps variables to elements of a universe, and the members of the universe probably will not be terms of the language we are using. So, if we replace a variable in a formula by what an assignment maps it to, we will not get a formula of our original language as a result. Here, we can see the reason of the Gerard's language extension. In the extended language we can describe the ground atoms, and examine the interpretations over the given universe with the help of a semantic tree. Of course, besides the elements of $\mathcal{P}$, we must interpret all the elements of $\mathcal{F}$ for a complete interpretation if $\mathcal{F} \neq \emptyset$.

It is inconvenient and impossible to consider all interpretations over all universes. It would be nice if we could construct a special universe such that we would have to take into account only the interpretations over that universe.

Definition 3.1. A Herbrand universe for a first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$ ($\mathcal{C} \neq \emptyset$) is the set of closed terms generated from function symbols of $\mathcal{F}$ and constant symbols of $\mathcal{C}$.

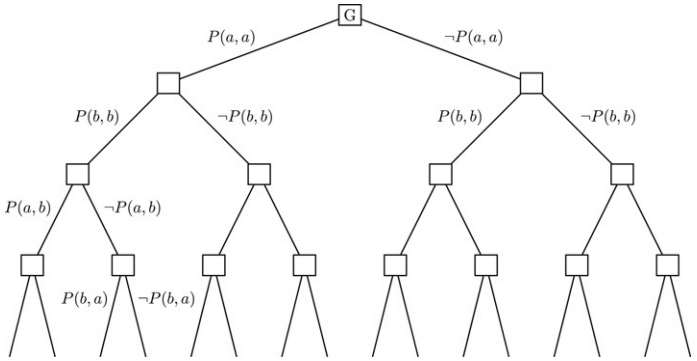

Fig. 1. Semantic tree.

Observe that, the members of the Herbrand universe are ground terms of $\mathcal{L}$ and at the same time names of universe elements. Here, the names of the elements depend on the language. Listing the members of the Herbrand universe determines an interpretation for the function symbols: naming the elements of the universe with $h_1, h_2, \ldots$ we get an interpretation for the function symbols over the set $\{h_1, h_2, \ldots\}$.

The following theorem is important because it traces back the examination of clauses of $\langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$ to the examination of formulae of $\langle \mathcal{P}, \emptyset, \mathcal{C} \rangle$ over the Herbrand universe. (The clauses are closed formulae in form $\forall x_1 \forall x_2 \cdots \forall x_n A$ where A is a disjunction of atoms and negations of atoms.)

Theorem 3.1. *A set of clauses $\mathcal{S}$ of $\mathcal{L}$ is unsatisfiable if and only if the set of all ground instances from the Herbrand universe of the clauses in $\mathcal{S}$ is unsatisfiable.*

This result led to the development of the so-called ground resolution calculus. The resolution rule for ground clauses is

$$\frac{L_1 \vee \cdots \vee L_n \vee A \qquad \neg A \vee K_1 \vee \cdots \vee K_m}{L_1 \vee \cdots \vee L_n \vee K_1 \vee \cdots \vee K_m}.$$

Here, L_i, K_j $(1 \leq i \leq n, 1 \leq j \leq m)$ are ground atoms or negation of them, and A is a single atom. A ground resolution derivation out of set $\mathcal{S}_0$ of ground clauses is a sequence $\mathcal{S}_0, \mathcal{S}_1, \ldots$ of sets of ground clauses such that for each $k \geq 1$, $\mathcal{S}_{k+1}$ is obtained by the application of this resolution rule to some pair of clauses in $\mathcal{S}_k$. The ground resolution procedure terminates, when an $\mathcal{S}_k$ containing the so-called empty clause (with no atom) arises. Such an $\mathcal{S}_k$ is unsatisfiable.

A more general resolution rule forms the basis of the promised method for arbitrary clauses, which is more efficient than the straightforward method of enumerating ground instances of certain clauses described before. The principal idea behind this concept is that of unification. Unification is a process providing a systematic means of finding substitutions which give rise to sets of ground instances of clauses whose existence is guaranteed by Theorem 3.1. To describe such a substitution it can be necessary naming the universe elements, too.

Sets, whether finite or infinite, obeying the following conditions are of fundamental importance in the tableau calculus.

Definition 3.2. Consider the language $\mathcal{L} = \langle \mathcal{P}, Par \rangle$ with a universe $\mathcal{U}$. A set $\mathcal{H}$ of closed $\mathcal{U}$-formulae is called a first-order Hintikka set with respect to $\mathcal{U}$, provided $\mathcal{H}$ is a propositional Hintikka set (see [17]), and in addition:

(1) If $\forall x A \in \mathcal{H}$, then $A_u^x \in \mathcal{H}$ for every u in $\mathcal{U}$.
(2) If $\exists x A \in \mathcal{H}$, then $A_u^x \in \mathcal{H}$ for at least one element u in $\mathcal{U}$.

The next theorem connects syntax and semantics.

Theorem 3.2 (*Hintikka's Lemma*). *Every first-order Hintikka set $\mathcal{H}$ with respect to $\mathcal{U}$ is satisfiable over the universe $\mathcal{U}$.*

According to [17] the first-order tableau rules for the language $\langle \mathcal{P}, Par \rangle$ are the following:

$$\frac{\forall x\, A}{A_a^x} \quad \text{(for any } a \in Par\text{)}, \qquad \frac{\exists x\, A}{A_a^x} \quad \text{(for a critical } a \in Par\text{)}.$$

The second rule is a formalization of the next informal argument. Suppose that in the course of a proof, we have established $\exists x\, A$. Then, we can say, let a be the name for an element having the property A. Here, we can use only such a symbol that has not been assigned any role yet. Since the parameters of the language $\langle \mathcal{P}, Par \rangle$ are "uncommitted", a critical (so that a new) one is always available for this purpose. Here, we can see the motivation of introduction of parameters to the logic language.

The tableau rules provide to arise Hintikka sets with respect to the set of parameters as a universe on the open branches (branch without any complement formula pair) of any finished systematic tableau. The universe is a special one: the parameters are names of the universe elements again. From the Hintikka's lemma we have at once that in any finished systematic tableau, every open branch is satisfiable (over this universe).

In the case of the languages $\langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$ the first-order tableau rules are given in the next form [11]

$$\frac{\forall x\, A}{A_t^x} \quad \text{(for any term } t\text{)}, \qquad \frac{\exists x\, A}{A_y^x} \quad \text{(for a critical variable } y\text{)}.$$

An arbitrary term is substituted into the body of a universal formula and any variable is substituted into the body of an existential formula which is not free in any formula of the branch which is being extended. The rules provide only, that the set of formulae arising on an open branch is a Hintikka set in $\langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$. According to [11] this set is satisfiable over the set of equivalence classes of terms introduced by tableau rules.

4. Toward the unified way to prove completeness

Soundness and completeness of a deduction system show its suitability for the treatment of logic. In this case, the syntactic and semantic constructions of logic are equivalent. The definitions of soundness and completeness properties and their proofs depend on the deduction system itself.

A deduction system is called sound, whenever its decision problem is solved for a given set of formulae, then this formula set has a special semantic property. Let us list the soundness theorems (H) for the Hilbert system, (R) for the resolution calculus and (T) for the tableau calculus.

Theorem 4.1 (*Soundness*).

(H) *If a formula A is deducible from a set $\mathcal{S}$ of formulae (in the Hilbert system), then $\mathcal{S} \models A$.*
(R) *If the empty clause has resolution deduction from a set $\mathcal{S}$ of clauses, then $\mathcal{S}$ is unsatisfiable.*
(T) *If the tableau of a set $\mathcal{S}$ of formulae is closed, then $\mathcal{S}$ is unsatisfiable.*

To prove the soundness property of a deduction system is not hard. In every case, the key fact needed is the soundness of the deduction rules.

Completeness of a deduction system means that if a set of formulae has the semantic property given by soundness, then the calculus works successfully over that set of formulae. The completeness theorems for the above systems are as follows.

Theorem 4.2 (*Completeness*).

(H) *If $\mathcal{S} \models A$, then A is deducible from $\mathcal{S}$.*
(R) *If a set $\mathcal{S}$ of clauses is unsatisfiable, then the empty clause has resolution deduction from $\mathcal{S}$.*
(T) *If a set $\mathcal{S}$ of formulae is unsatisfiable, then the tableau of $\mathcal{S}$ is closed.*

By semantics, a set of formulae is either satisfiable or not. This semantic property divides the set Ω of sets of formulae into two disjoint parts. In a given calculus we can define a syntactic property for the sets of formulae dividing the set Ω into two disjoint parts, as well. This is generally called consistency (inconsistency) property. The definition of the (in)consistency property depends on the deduction system.

Definition 4.1. (H) $\mathcal{S}$ is inconsistent if A and $\neg A$ are deducible from $\mathcal{S}$.
(R) A set S of clauses is inconsistent if the empty clause has resolution deduction from the set S.
(T) The set $\mathcal{S}$ of formulae is inconsistent if the tableau of $\mathcal{S}$ is closed.

It is obvious that, the soundness and completeness properties of a deduction system can be expressed by the (in)consistency property: soundness requires that if $\mathcal{S}$ is inconsistent, then $\mathcal{S}$ must be unsatisfiable, and completeness requires that if $\mathcal{S}$ is unsatisfiable, then $\mathcal{S}$ must be inconsistent. Then, completeness of a deduction system is proved by showing that the consistency–inconsistency (syntactic) properties and the satisfiability–unsatisfiability (semantic) properties divide into the same two parts the set Ω of sets of formulae.

Let Γ denote any property of formula sets which is of finite character. It means that a set $\mathcal{S}$ has the property Γ if and only if all finite subset of $\mathcal{S}$ have the property Γ.

Definition 4.2. In the language $\mathcal{L} = \langle \mathcal{P}, Par \rangle$, a property Γ of finite character is called an analytic consistency property for first-order logic if Γ is an analytic consistency property for propositional logic [17,18], and if for every set $\mathcal{S}$ of formulae of $\mathcal{L}$ having the property Γ the following conditions hold:

(1) If $\forall x A \in \mathcal{S}$, then for every $a \in Par$, $\mathcal{S} \cup \{A_a^x\}$ has the property Γ.
(2) If $\exists x A \in \mathcal{S}$, then $\mathcal{S} \cup \{A_a^x\}$ has the property Γ, if $a \in Par$ does not occur in $\mathcal{S}$.

An important example of an analytic consistency property is the consistency property in the tableau method.

Theorem 4.3 (*Unifying Principle*). *If Γ is an analytic consistency property, $\mathcal{S}$ is a set of parameter-free formulae in $\langle \mathcal{P}, Par \rangle$, and $\mathcal{S}$ has the property Γ, then $\mathcal{S}$ is satisfiable.*

It is clear that a set $\mathcal{S}$ of formulae having an analytic consistency property can be embedded into a Hintikka set with respect to the set of parameters as a universe, so it is satisfiable.

Consequently, instead of proving the different completeness theorems it is sufficient to test whether the consistency property in a given deduction system is an analytic consistency property. If the consistency property is an analytic consistency property, then this fact is equivalent to the completeness of this deduction system. By this result we get a unified method to treat the completeness problem of a deduction system in the first-order logic.

There is another important application of the unifying principle. It is easily verified that the following property Γ is an analytic consistency property: Let a set $\mathcal{S}$ of parameter-free formulae in $\langle \mathcal{P}, Par \rangle$ have the property Γ, if all of its finite subsets are satisfiable. Using Theorem 4.3, we get the so-called compactness theorem for first-order logic: If every finite subset of $\mathcal{S}$ is satisfiable, so is $\mathcal{S}$.

5. Applications

Many applications of logic, mainly in the artificial intelligence, are related to the deduction systems. In these applications, naming the universe elements is inevitable. To solve a problem by a deduction system the following steps are executed.

(I) The first task is to create an ideal world, a mathematical model for the problem, in which the classical logic can be used to reason correctly. (Whether the model accurately reflects the real world is a separate issue.) In this model we have a universe. Our ideal world is characterized by operations and relations on this universe.
(II) The second task is to find a description language for the ideal world. The extralogical part of the alphabet consists of predicate and function symbols identifying the relations and operations of the model. For describing assertions about universe elements, it is necessary to introduce constant symbols naming them.
(III) This language is suitable for formalization of the original problem. The result of formalization is often a finite set of formulae (premises) and a formula (conclusion) in the language. The third task is to test whether the conclusion is a consequence of premises. The actually used deduction system tries to give the answer with solving its own decision problem according to the original one.

In the case of using the resolution calculus (for example in a Prolog system) the Herbrand universe is used, where the constant symbols of the language are the constant elements of the Herbrand universe. If there are function symbols in the language, then the ground terms are also elements of the Herbrand universe. As the problem has the originally fixed universe, then the Herbrand universe shows an interpretation of function symbols

over the constants. If a method, different to the resolution, is used, then the introduced constant symbols are the names of the universe elements.

Another field of applications is the relational database theory. The description and query language Datalog uses a logic language to denominate the relations and describe their connections. The language has a tool to name the data elements, too. Therefore, while Datalog executes the steps of resolution deduction, it can use the relational calculus instead of unification.

6. Conclusion

In this paper, we proved that, the applications require from the description language the ability to name the elements of their domain. So, the possibility of an interpretation-dependent extension of first-order languages is an important issue in logic. The interpretation-dependent extension of a language involves that, the models of the extended language are models on the set of the introduced constant symbols. This means that, the set of models of the extended language is a subset of the set of the models of the original language.

Some trends in logic offers the possibility to fix a universe first and extends the alphabet of the logic language by the names of the universe elements. In this case, the formulae become pre-interpreted before the interpretation. This means the constant (and the parameter) symbols can be substituted by these names in the formulae. The Gerard's language and the language $\langle \mathcal{P}, Par \rangle$ have come off in principle just for that reason.

References

[1] L. Couturat, Opuscules et Fragments inédits de Leibniz, extracts des manuscripts de la Bibliothéque royale de Hanovre, Paris, 1903.
[2] G. Frege, Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, Halle, 1879.
[3] G. Peano, Arithmetices Principia, Nova Methodo Exposita, Fratres Bocca, Turin, 1889.
[4] E. Mendelson, Introduction to Mathematical Logic, D. Van Nostrand Company, Inc., 1964.
[5] S.C. Kleene, Mathematical Logic, John Wiley and Sons, New York, 1967.
[6] L. Kalmár, A matematika alapjai I-IV., JATE egyetemi jegyzet, (University Lecture Notes) Szeged, 1971 (in Hungarian).
[7] Sz. Buzási, A. Dragálin, KLTE egyetemi jegyzet (University Lecture Notes) Debrecen, 1986 (in Hungarian).
[8] M. Davis, First Order Logic, in: Handbook of Logic in Artificial Intelligence and Logic Programming, Clarendon Press, Oxford, 1993.
[9] K. Pásztorné Varga, A matematikai logika és alkalmazásai, ELTE egyetemi jegyzet (University Lecture Notes) Budapest, 1989 (in Hungarian).
[10] K. Pásztorné Varga, Matematikai logika alkalmazásokhoz, ELTE egyetemi jegyzet (University Lecture Notes) Budapest, 1997 (in Hungarian).
[11] C. Bell, M. Machover, A Course in Mathematical Logic, Elsevier North-Holland, Amsterdam, 1977.
[12] J.-Y. Girard, Proof Theory and Logical Complexity, vol. 1, Bibliopolis, Napoli, 1987.
[13] U. Schöning, Logic für Informatiker, Spektrum Akademischer Verlag, Heidelberg, 1995.
[14] A. Nerode, R.A. Shore, Logic for Applications, Springer-Verlag, New York, 1997.
[15] R. Socher-Ambrosius, P. Johann, Deduction Systems, Springer-Verlag, New York, 1997.
[16] Yu.L. Ershov, E.A. Palutyin, Mathematical Logic, Mir Publishers, Moszkva, 1984 (in Russian).
[17] R.M. Smullyan, First-Order Logic, Springer-Verlag, Berlin, 1968.
[18] M. Fitting, First-order Logic and Automated Theorem Proving, Springer-Verlag, New York, 1996.